Mining Tasks from the Web Anchor Text Graph: MSR Notebook Paper for the TREC 2015 Tasks Track

Paul N. Bennett
Microsoft Research
Redmond, USA
pauben@microsoft.com

Ryen W. White
Microsoft Research
Redmond, USA
ryenw@microsoft.com

January 31, 2016

1 Introduction

Users may have a variety of tasks that give rise to issuing a particular query. The goal of the Tasks Track at TREC 2015 was to identify all aspects or subtasks of a user's task as well as the documents relevant to the entire task. This was broken into two parts: (1) *Task Understanding* which judged relevance of key phrases or queries to the original query (relative to a likely task that would have given rise to both); (2) *Task Completion* which performed document retrieval and measured usefulness to any task a user with the query might be peforming through either a completion measure that uses both relevance and usefulness criteria or more simply through an *ad hoc* retrieval measure of relevance alone. We submitted a run in the Task Understanding track. In particular, since the anchor text graph has proven useful in the general realm of query reformulation [2], we sought to quantify the value of extracting key phrases from anchor text in the broader setting of the task understanding track.

Given a query, our approach considers a simple method for identifying a relevant and diverse set of key phrases related to the possible tasks that might have given rise to the query. In particular, given the effectiveness of sessions for producing query suggestions as well as the fact that sessions tend to be both topically coherent and cohesive with respect to a task, we investigated the effectiveness of mining session co-occurrence data. For a search engine log, session boundaries can be defined in the typical way but to operate over the anchor text graph, we need some notion of a "session". We adopt the suggestion of Dang & Croft [2] and treat different links pointing to the same document as belonging to the same "session". The basic assumption is that the anchor text of two links pointing to the same document are related via the common reference. Note that this assumption is based on the *destination* URL of the link being the same.

Given a query, we then find matching seed candidates (link text from the web graph or queries over search logs) and expand to related candidate key phrases via this session assumption. The final ranking is based on a combination of session count and the similarity of a link to the query. Additionally we perform several types of filtering that prevent over-expanding the set of related queries. We refer to the method as "having coverage" if the method was able to find a matching seed – since this is a necessary step to producing any candidates based on co-occurrence.

Empirical results demonstrate generally good performance for the method when it finds a matching seed. In particular, of the 34 topics judged for the Query Understanding track, our method had coverage 62% of the time (21 topics). When the method has coverage, the suggested key phrases are above the mean performance (by nearly every measure reported) 2/3 times and the best performer 1/3 times. Given it's simplicity and availability to nearly all participants as well as the fact that coverage can be detected before submission, it is a promsing candidate for future investigation in the track. We now describe the method and results more fully before summarizing.

2 Query Understanding Task

This section deals with the problem investigated in the Query Understanding task of the track. That is, how can we effectively identify all key phrases or alternate queries that might be involved in any task a user might have which would give rise to the observed query for a topic?

2.1 Approach Overview

To provide a brief overview, we remind the reader that our goal is to investigate the effectiveness of basic session co-occurrence for the task understanding task. For search logs, taking the commonly accepted session boundaries of a period of user activity demarcated by 30 minutes of inactivity is straightforward, but to apply the same approach to anchor text we need a notion of session. Like others [2] for the anchor text graph, we define the text of two links to co-occur in a "session" if the links point to the same destination URL. Given a query, we find matching seed candidates (link text

from the web graph or queries over search logs) using a soft matching. These seed candidates are further expanded to all queries/links co-occurring in a session. The candidates are pruned to filter out any globally frequent queries across all sessions, extremely long queries, or candidates that do not provide a minimum similarity to the original query. The final ranking is a simple product of the session count in matched sessions and a similarity of the candidate with the original query. We now describe the method in more detail.

2.1.1 ClueWeb12 Anchor Text Graph

For most interested participants an extraction of the anchor text graph of ClueWeb12 is easily available at http://wwwhome.ewi.utwente.nl/~hiemstra/2013/anchor-text-for-clueweb12.html [3], however we chose[1] to produce an anchor text graph directly from the ClueWeb12 dataset.[2]

In particular, we used the publically available HTMLAgilityPack[3] v. 1.4.9.0. Similar to Hiemstra & Hauff [3] we only process documents whose html size was less than 50K bytes and we discard "javascript" or "mailto" links. Additionally, we attempt to retain[4] only links where the destination is to an external site. We do this since internal links are often either navigational or assume context. That is, anchor text of "bothell campus" on a "University of Washington" page is likely pointing at the home page for the "University of Washington at Bothell" but the simple descriptor "bothell campus" would not be a high quality keyphrase suggestion absent the context. Additionally, we also only retain links whose destination URL resolves to a document in ClueWeb12 (i.e. we discard links pointing out of ClueWeb12). This was done under the assumption that documents within ClueWeb12 may have better crawl coverage across multiple incoming links.

2.1.2 Normalizing Queries to Filter Phrases

For the text in each of the query fields in the topic xml file, queries were first normalized by removing multiple whitespaces and converting all characters to lower case. After this, we removed stopwords from each of the query text. Rather than stopword lists that are developed based on frequency alone, we used a list of English *function* words, *e.g.*, "a", "about", "the", "to", "who", *etc.* While function words are correlated with frequency, anecdotally we found other published stopword lists developed by frequency alone or by taking the most frequent words in our corpora to be too aggressive. Investigating the impact of this choice is an interesting direction for future work.

[1] Unfortunately the anchor text distribution is via peer-to-peer software the use of which is procedurally complicated at our organization.

[2] See http://www.lemurproject.org/clueweb12.php/ for more on ClueWeb12.

[3] http://htmlagilitypack.codeplex.com/

[4] Many types of relative links might not be detected as well as sites which appear to be different by the URL but are owned by the same organization.

As our list of English function words, we took the list of 221 words published by Cook [1] and available for download at http://homepage.ntlworld.com/vivian.c/Words/StructureWordsList.htm. For a topic, t, with a particular query q_t, we refer to the output after query normalization and stopword removal as a filter phrase, f_t.

2.1.3 Compute Globally Frequent Candidates

For any query suggestion method, any co-occurrence approach has to deal with globally frequent items that co-occur independently. To deal with this, we compute the top 1K most frequent texts based on the input. For example, for the ClueWeb12 anchor text graph, the top four most common anchor texts are: "next", "permalink", "prev", and "read more". These top 1K globally frequent candidates will be pruned out later.

2.1.4 Matching Filter Phrases to Seed Candidates

After normalizing queries to filter phrases as described in Section 2.1.2, for each topic a filter phrase f_t is considered to match a candidate seed, c_t, if the candidate is a superset of f_t. That is the candidate seed, c_t contains at least all of the words in f_t. Because the filter phrase, f_t, is never bigger than the original query, q_t. This means the filter phrase will match seeds that exactly match the query, partially overlap with the query (as long as all non-function words overlap), and are supersets of the query. The intuition behind adding supersets is since the goal is to identify all possible tasks that might lead to the query, users' queries or anchor text links often contain extra words that are specific to some particular task. An interesting line of future work is to separate the contribution of exact matching seeds (with and without order), overlapping matching seeds, and superset seeds. This set of matching sessions $\mathcal{S}_t$, which contain a candidate seed match, is the basis for the remainder of our computation for a topic t.

2.1.5 Expanding to Related Candidates

From the matching sessions containing a matching seed, we expand to related candidates by removing all globally frequent candidates and simply counting the number of sessions in $\mathcal{S}_t$ each remaining query occurs in. As in the query suggestion literature, future work could consider multiple rounds of expansion or a weighted random walk.

2.1.6 Filtering and Similarity Weighting

The expansion to related candidates based on co-occurrence has several types of common failures. To be conservative and attempt to eliminate these failures, we require a candidate to have overlap with the filter phrase for a topic and meet a length restriction (very long texts will tend to match

spuriously). In particular, for every topic t and candidate keyphrase k_t and filter phrase f_t:

1. k_t is discarded if $\cos(k_t, f_t) \leq 0$ (i.e. there must be at least one word overlap).

2. k_t is discarded if its length is longer than a multiple of f_t, i.e. if $k_t > f_t L$. We choose $L = 4$ to allow a generous but not extreme upper bound. This condition gets rid of extremely long candidates – usually pastes in search logs or bad tag closures in anchor text.

To produce the final score, it's intuitive that not only should the frequency of occurrence in a matching session matter, but the candidate's similarity to the query is likely important. To account for this, we weight the count of occurrences in matching sessions, s_{k_t}, by the similarity to the filter phrase before normalization. More precisely, we first

1. Scale the matching session occurrence count for the keyphrase by the similarity to the filter phrase: $s_{k_t}^{\text{unnorm}} = \cos(k_t, f_t) \cdot s_{k_t}$.

2. Normalize the final score by the max in the topic: $s_{k_t}^{\text{final}} = s_{k_t}^{\text{unnorm}} / \max_{k_t} s_{k_t}^{\text{unnorm}}$. This simply scales the final score to the $[0, 1]$ range and does not alter the final ranking.

2.2 Evaluation

Table 1 reports the mean across all topics differences from the per topic mean by each performance measure (i.e. positive means above mean overall and negative below mean). Table 2 reports similar values but only over the 21 of the 34 judged topics where the method had coverage. In the remaining 13 a matching seed was not found. Table 3 reports of the 21 times when the method had coverage, the number of topics where the method the best performer or above average.

As can be seen, overall the method falls below the mean across all topics, but when taking Table 2 into account, this is because the method sometimes lacks coverage. Since this can be easily detected, the potential of this method for use in combination with other techniques is represented by its performance when it has coverage. In those cases, the mean across topics is quite positive with (as seen in Table 3) the method performing above the mean 2/3 times and obtaining the best performance 1/3 times that it has coverage. Overall, this speaks well to the potential for combining this method with techniques used by other participants.

3 Conclusions

We described a simple approach that can be equally applied to either search logs or the anchor text graph for finding keyphrases for related tasks to a query. The method relies on a simple matching procedure to find starting seeds and uses either common destinations in the anchor text graph or session co-occurrence in a search log to find related candidates. Simple steps of filtering are applied to remove globally frequent candidates as well as candidate keyphrases that have no similarity to the core of the original query.

Overall, empirical results demonstrate generally good performance for the method when it finds a matching seed. In particular, of the 34 topics judged for the Query Understanding track, our method had coverage 62% of the time (21 topics). When the method has coverage, the suggested key phrases are above the mean performance (by nearly every measure reported) 2/3 times and the best performer 1/3 times. Given its simplicity and availability to nearly all participants as well as the fact that coverage can be detected before submission, it is a promising candidate for future investigation in the track.

References

[1] V. Cook. Designing a basic parser for call. *CALICO Journal*, 6(1):50–67, 1988. http://homepage.ntlworld.com/vivian.c/Writings/Papers/CalicoPaper88.htm.

[2] V. Dang and W. Croft. Query reformulation using anchor text. In *WSDM 2010*, pages 41–50, 2010.

[3] D. Hiemstra and C. Hauff. Mirex: Mapreduce information retrieval experiments. Technical Report CTIT Technical Report TR-CTIT-10-15, Centre for Telematics and Information Technology, University of Twente, 2010. ISSN 1381-3625.

ERR-IA@10	ERR-IA@20	ERR-IA@1000	α-nDCG@10	α-nDCG@20	α-nDCG@1000
-0.0154	-0.0219	-0.0232	-0.0177	-0.0446	-0.0681

Table 1: Overall Average of Difference from Mean Topic.

ERR-IA@10	ERR-IA@20	ERR-IA@1000	α-nDCG@10	α-nDCG@20	α-nDCG@1000
0.0959	0.0894	0.0882	0.1126	0.0871	0.0632

Table 2: Overall Average of Difference from Mean Topic when Coverage.

	ERR-IA@10	ERR-IA@20	ERR-IA@1000	α-nDCG@10	α-nDCG@20	α-nDCG@1000
Max	7	7	7	7	6	6
$>$ Mean	15	15	15	15	15	14

Table 3: Times best and better than mean when coverage.